4.4.1 Administración del espacio en disco

Por lo general los archivos se almacenan en disco, así que la administración del espacio en disco es una cuestión importante para los diseñadores de sistemas de archivos. Hay dos estrategias generales posibles para almacenar un archivo de n bytes: se asignan n bytes consecutivos de espacio en disco o el archivo se divide en varios bloques (no necesariamente) contiguos. La misma concesión está presente en los sistemas de administración de memoria, entre la segmentación pura y la paginación. Como hemos visto, almacenar un archivo como una secuencia contigua de bytes tiene el problema obvio de que si un archivo crece, probablemente tendrá que moverse en el disco. El mismo problema se aplica a los segmentos en memoria, excepto que la operación de mover un segmento en memoria es rápida, en comparación con la operación de mover un archivo de una posición en el disco a otra. Por esta razón, casi todos los sistemas de archivos dividen los archivos en bloques de tamaño fijo que no necesitan ser adyacentes.

Tamaño de bloque.

Una vez que se ha decidido almacenar archivos en bloques de tamaño fijo, surge la pregunta acerca de qué tan grande debe ser el bloque. Dada la forma en que están organizados los discos, el sector, la pista y el cilindro son candidatos obvios para la unidad de asignación (aunque todos ellos son dependientes del dispositivo, lo cual es una desventaja). En un sistema de paginación, el tamaño de la página también es uno de los principales contendientes. Tener un tamaño de bloque grande significa que cada archivo (incluso un archivo de 1 byte) ocupa un cilindro completo. También significa que los pequeños archivos desperdician una gran cantidad de espacio en disco. Por otro lado, un tamaño de bloque pequeño significa que la mayoría de los archivos abarcarán varios bloques y por ende, necesitan varias búsquedas y retrasos rotacionales para leerlos, lo cual reduce el rendimiento. Por ende, si la unidad de asignación es demasiado grande, desperdiciamos espacio; si es demasiado pequeña, desperdiciamos tiempo. Para hacer una buena elección hay que tener cierta información sobre la distribución de los tamaños de archivo. Tanenbaum y colaboradores (2006) estudiaron la distribución de los tamaños de archivo en el Departamento de Ciencias Computacionales de una gran universidad de investigación (la VU) en 1984 y después en el 2005, así como en un servidor Web comercial que hospedaba a un sitio Web político (www.electoral-vote.com). Los resultados se muestran en la figura 4-20, donde para cada tamaño de archivo de una potencia de dos, se lista el porcentaje de todos los archivos menores o iguales a éste para cada uno de los tres conjuntos de datos. Por ejemplo, en el 2005 el 59.13% de todos los archivos en la VU eran de 4 KB o menores y el 90.84% de todos los archivos.

Longitud	VU 1984	VU 2005	Web
1	1.79	1.38	6.67
2	1.88	1.53	7.67
4	2.01	1.65	8.33
8	2.31	1.80	11.30
16	3.32	2.15	11.46
32	5.13	3.15	12.33
64	8.71	4.98	26.10
128	14.73	8.03	28.49
256	23.09	13.29	32.10
512	34.44	20.62	39.94
1 KB	48.05	30.91	47.82
2 KB	60.87	46.09	59.44
4 KB	75.31	59.13	70.64
8 KB	84.97	69.96	79.69

Longitud	VU 1984	VU 2005	Web
16 KB	92.53	78.92	86.79
32 KB	97.21	85.87	91.65
64 KB	99.18	90.84	94.80
128 KB	99.84	93.73	96.93
256 KB	99.96	96.12	98.48
512 KB	100.00	97.73	98.99
1 MB	100.00	98.87	99.62
2 MB	100.00	99.44	99.80
4 MB	100.00	99.71	99.87
8 MB	100.00	99.86	99.94
16 MB	100.00	99.94	99.97
32 MB	100.00	99.97	99.99
64 MB	100.00	99.99	99.99
128 MB	100.00	99.99	100.00

Figura 4-20. Porcentaje de archivos menores que un tamaño dado (en bytes).

eran de 64 KB o menores. El tamaño de archivo promedio era de 2475 bytes. Tal vez a algunas personas este tamaño tan pequeño les parezca sorprendente. ¿Qué conclusiones podemos obtener de estos datos? Por una parte, con un tamaño de bloque de 1 KB sólo entre 30 y 50% de todos los archivos caben en un solo bloque, mientras que con un bloque de 4 KB, el porcentaje de archivos que caben en un bloque aumenta hasta el rango entre 60 y 70%. Los demás datos en el artículo muestran que con un bloque de 4 KB, 93% de los bloques de disco son utilizados por 10% de los archivos más grandes. Esto significa que el desperdicio de espacio al final de cada pequeño archivo no es muy importante, debido a que el disco se llena por una gran cantidad de archivos grandes (videos) y la cantidad total de espacio ocupado por los pequeños archivos es insignificante. Incluso si se duplica el espacio que 90% de los archivos más pequeños ocuparían, sería insignificante. Por otro lado, utilizar un bloque pequeño significa que cada archivo consistirá de muchos bloques. Para leer cada bloque normalmente se requiere una búsqueda y un retraso rotacional, por lo que la acción de leer un archivo que consista de muchos bloques pequeños será lenta. Como ejemplo, considere un disco con 1 MB por pista, un tiempo de rotación de 8.33 mseg y un tiempo de búsqueda promedio de 5 mseg. El tiempo en milisegundos para leer un bloque de k bytes es entonces la suma de los tiempos de búsqueda, de retraso rotacional y de transferencia:

$$5 + 4.165 + \left(\frac{k}{1000000}\right) * 8.33$$

La curva sólida de la figura 4-21 muestra la velocidad de los datos para dicho disco como una función del tamaño de bloque. Para calcular la eficiencia del espacio, necesitamos hacer una suposición acerca del tamaño de archivo promedio. Por simplicidad, vamos a suponer que todos los archivos son de 4 KB. Aunque este número es un poco más grande que los datos medidos en la VU, es probable que los estudiantes tengan más archivos pequeños de los que estarían presentes en un centro de datos corporativo, por lo que podría ser una mejor aproximación en general. La curva punteada de la figura 4-21 muestra la eficiencia del espacio como una función del tamaño de bloque.

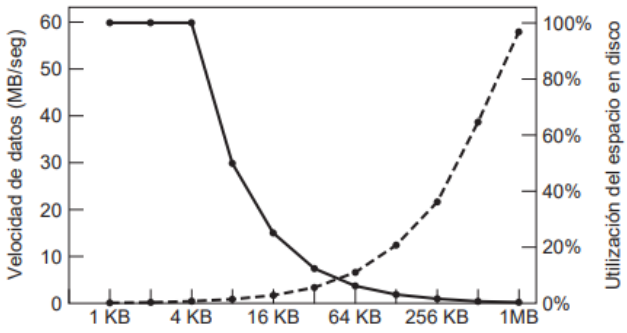

Figura 4-21. La curva sólida (escala del lado izquierdo) da la velocidad de datos del disco. La curva punteada (escala del lado derecho) da la eficiencia del espacio de disco. Todos los archivos son de 4 KB.

Las dos curvas se pueden comprender de la siguiente manera. El tiempo de acceso para un bloque está completamente dominado por el tiempo de búsqueda y el retraso rotacional, dado que se van a requerir 9 mseg para acceder a un bloque, entre más datos se obtengan será mejor. En consecuencia, la velocidad de datos aumenta casi en forma lineal con el tamaño de bloque (hasta que las transferencias tardan tanto que el tiempo de transferencia empieza a ser importante). Ahora considere la eficiencia del espacio. Con archivos de 4 KB y bloques de 1 KB, 2 KB o 4 KB, los archivos utilizan el 4, 2 y 1 bloques, respectivamente, sin desperdicio. Con un bloque de 8 KB y archivos de 4 KB, la eficiencia del espacio disminuye hasta el 50% y con un bloque de 16 KB disminuye hasta 25%. En realidad, pocos archivos son un múltiplo exacto del tamaño de bloque del disco, por lo que siempre se desperdicia espacio en el último bloque de un archivo. Sin embargo, lo que muestran las curvas es que el rendimiento y el uso del espacio se encuentran inherentemente en conflicto. Los bloques pequeños son malos para el rendimiento, pero buenos para el uso del espacio en el disco. Para estos datos no hay un compromiso razonable disponible. El tamaño más cercano al punto en el que se cruzan las dos curvas es 64 KB, pero la velocidad de datos es de sólo 6.6 MB/seg y la eficiencia del espacio es de aproximadamente 7%; ninguno de estos dos valores es bueno. Históricamente, los sistemas de archivos han elegido tamaños en el rango de 1 KB a 4 KB, pero con discos que ahora exceden a 1 TB, podría ser mejor incrementar el tamaño de bloque a 64 KB y aceptar el espacio en disco desperdiciado. El espacio en disco ya no escasea en estos días. En un experimento por ver si el uso de archivos en Windows NT era muy distinto al de UNIX, Vogels hizo mediciones en los archivos de la universidad Cornell University (Vogels, 1999). Él observó que el uso de archivos en NT es más complicado que en UNIX. Escribió lo siguiente:

“Cuando escribimos unos cuantos caracteres en el editor de texto bloc de notas (notepad), al guardar esto a un archivo se activarán 26 llamadas al sistema, incluyendo 3 intentos de apertura fallidos, 1 sobrescritura de archivo y 4 secuencias adicionales de abrir y cerrar.”

Sin embargo, observó un tamaño promedio (ponderado por uso) de archivos sólo leídos de 1 KB, de archivos sólo escritos de 2.3 KB y de archivos leídos y escritos de 4.2 KB. Dadas las distintas técnicas de medición de los conjuntos de datos y el año, estos resultados son sin duda compatibles con los resultados de la VU.

Registro de Bloques libres

Una vez que se ha elegido un tamaño de bloque, la siguiente cuestión es cómo llevar registro de los bloques libres. Hay dos métodos utilizados ampliamente, como se muestra en la figura 4-22. El primero consiste en utilizar una lista enlazada de bloques de disco, donde cada bloque contiene tantos números de bloques de disco libres como pueda. Con un bloque de 1 KB y un número de bloque de disco de 32 bits, cada bloque en la lista de bloques libres contiene los números de 255 bloques libres (se requiere una ranura para el apuntador al siguiente bloque). Considere un disco de 500 GB, que tiene aproximadamente 488 millones de bloques de disco. Para almacenar todas estas direcciones a 255 por bloque, se requiere una cantidad aproximada de 1.9 millones de bloques. En general se utilizan bloques libres para mantener la lista de bloques libres, por lo que en esencia el almacenamiento es gratuito.

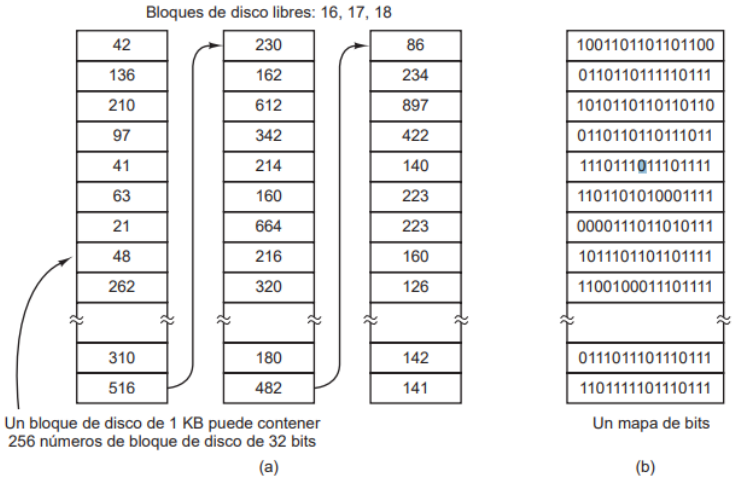

Figura 4-22. (a) Almacenamiento de la lista de bloques libres en una lista enlazada. (b) Un mapa de bits.

La otra técnica de administración del espacio libre es el mapa de bits. Un disco con n bloques requiere un mapa de bits con n bits. Los bloques libres se representan mediante 1s en el mapa, los bloques asignados mediante 0s (o viceversa). Para nuestro disco de 500 GB de ejemplo, necesitamos488 millones de bits para el mapa, que requieren poco menos de 60,000 bloques de 1 KB para su almacenamiento. No es sorpresa que el mapa de bits requiera menos espacio, ya que utiliza 1 bit por bloque, en comparación con 32 bits en el modelo de la lista enlazada. Sólo si el disco está casi lleno (es decir, que tenga pocos bloques libres) es cuando el esquema de la lista enlazada requiere menos bloques que el mapa de bits. Si los bloques libres tienden a presentarse en series largas de bloques consecutivos, el sistema de la lista de bloques libres se puede modificar para llevar la cuenta de las series de bloques, en vez de bloques individuales. Una cuenta de 8, 16 o 32 bits se podría asociar con cada bloque dando el número de bloques libres consecutivos. En el mejor caso, un disco prácticamente vacío podría representarse mediante dos números: la dirección del primer bloque libre seguida de la cuenta de bloques libres. Por otro lado, si el disco se fragmenta demasiado, es menos eficiente llevar el registro de las series que de los bloques individuales, debido a que no sólo se debe almacenar la dirección, sino también la cuenta. Esta cuestión ilustra un problema con el que los diseñadores de sistemas operativos se enfrentan a menudo. Hay varias estructuras de datos y algoritmos que se pueden utilizar

para resolver un problema, pero para elegir el mejor se requieren datos que los diseñadores no tienen y no tendrán sino hasta que el sistema se opere y se utilice con frecuencia. E incluso entonces, tal vez los datos no estén disponibles. Por ejemplo, nuestras propias mediciones de los tamaños de archivo en la VU en 1984 y 1995, los datos del sitio Web y los datos de Cornell son sólo cuatro muestras. Aunque es mejor que nada, tenemos una idea muy vaga acerca de si también son representativos de las computadoras domésticas, las computadoras empresariales, las computadoras gubernamentales y otras. Con algo de esfuerzo podríamos haber obtenido un par de muestras de otros tipos de computadoras, pero incluso así sería imprudente extrapolar a todas las computadoras de los tipos que se midieron. Regresando por un momento al método de la lista de bloques libres, sólo hay que mantener un bloque de apuntadores en la memoria principal. Al crear un archivo, los bloques necesarios se toman del bloque de apuntadores. Cuando este bloque se agota, se lee un nuevo bloque de apuntadores del disco. De manera similar, cuando se elimina un archivo se liberan sus bloques y se agregan al bloque de apuntadores en la memoria principal. Cuando este bloque se llena, se escribe en el disco. Bajo ciertas circunstancias, este método produce operaciones de E/S de disco innecesarias. Considere la situación de la figura 4-23(a), donde el bloque de apuntadores en memoria tiene espacio para sólo dos entradas más. Si se libera un archivo de tres bloques, el bloque de apuntadores se desborda y tiene que escribirse en el disco, con lo cual se produce la situación de la figura 4.23(b). Si ahora se escribe un archivo de tres bloques, se tiene que volver a leer el bloque completo de apuntadores del disco, lo cual nos regresa a la figura 4-23(a). Si el archivo de tres bloques que se acaba de escribir era temporal, al liberarlo se necesitará otra escritura de disco para escribir el bloque completo de apuntadores de vuelta en el disco. En resumen, cuando el bloque de apuntadores está casi vacío, una serie de archivos temporales de un tiempo corto de vida puede producir muchas operaciones de E/S de disco. Un método alternativo que evita la mayor parte de estas operaciones de E/S de disco es dividir el bloque completo de apuntadores. Así, en vez de pasar de la figura 4-23(a) a la figura 4-23(b), pasamos de la figura 4-23(a) a la figura 4-23(c) cuando se liberan tres bloques. Ahora el sistema puede manejar una serie de archivos temporales sin realizar ninguna operación de E/S. Si el bloque en memoria se llena, se escribe en el disco y se lee el bloque medio lleno del disco. La idea aquí es mantener la mayor parte de los bloques de apuntadores en el disco llenos (para minimizar el uso del disco), pero mantener el que está en memoria lleno a la mitad, para que pueda manejar tanto la creación como la remoción de archivos sin necesidad de operaciones de E/S de disco en la lista de bloques libres. Con un mapa de bits también es posible mantener sólo un bloque en memoria, usando el disco para obtener otro sólo cuando el primero se llena o se vacía. Un beneficio adicional de este método es que, al realizar toda la asignación de un solo bloque del mapa de bits, los bloques de disco estarán cerca uno del otro, con lo cual se minimiza el movimiento del brazo del disco. Como el mapa de bits es una estructura de datos de tamaño fijo, si el kernel está paginado (parcialmente), el mapa de bits puede colocarse en memoria virtual y hacer que se paginen sus páginas según se requiera.

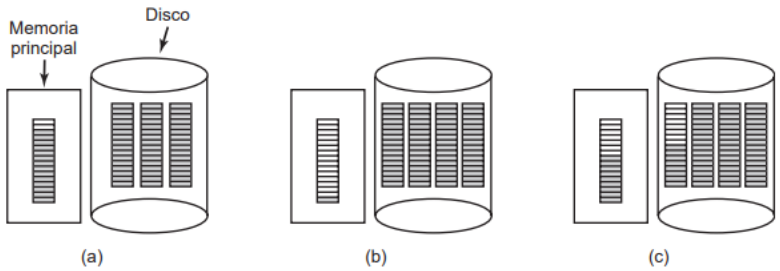

Figura 4-23. (a) Un bloque casi lleno de apuntadoras a bloques de disco libres en la memoria y tres bloques de apuntadores en el disco. (b) Resultado de liberar un archivo de tres bloques. (c) Una estrategia alternativa para manejar los tres bloques libres. Las entradas sombreadas representan apuntadores a bloques de disco libres.

Cuotas de disco

Para evitar que los usuarios ocupen demasiado espacio en disco, los sistemas operativos multiusuario proporcionan un mecanismo para imponer las cuotas de disco. La idea es que el administrador del sistema asigne a cada usuario una cantidad máxima de archivos y bloques y que el sistema operativo se asegure de que los usuarios no excedan sus cuotas. A continuación, describiremos un mecanismo común. Cuando un usuario abre un archivo, los atributos y las direcciones de disco se localizan y se colocan en una tabla de archivos abiertos en la memoria principal. Entre los atributos hay una entrada que indica quién es el propietario. Cualquier aumento en el tamaño del archivo se tomará de la cuota del propietario. Una segunda tabla contiene el registro de cuotas para cada usuario con un archivo actualmente abierto, aun si el archivo fue abierto por alguien más. Esta tabla se muestra en la figura 4-24. Es un extracto de un archivo de cuotas en el disco para los usuarios cuyos archivos están actualmente abiertos. Cuando todos los archivos se cierran, el registro se escribe de vuelta en el archivo de cuotas.

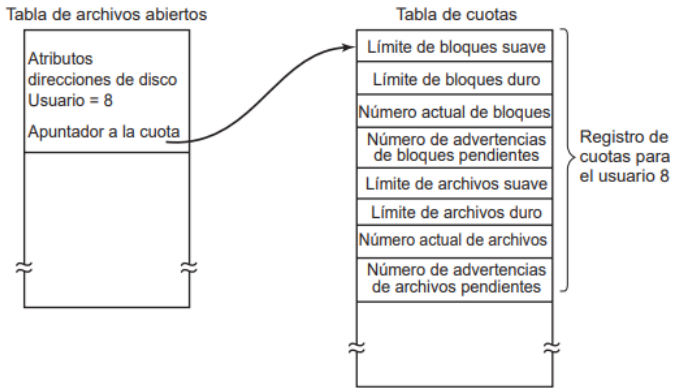

Figura 4-24. El registro de cuotas se lleva en una tabla de cuotas, usuario por usuario.

Cuando se crea una nueva entrada en la tabla de archivos abiertos, se introduce en ella un apuntador al registro de cuotas del propietario, para facilitar la búsqueda de los diversos límites. Cada vez que se agrega un bloque a un archivo se incrementa el número total de bloques que se cargan al propietario y se realiza una verificación con los límites duro y suave. Se puede exceder el límite suave, pero el límite duro no. Un intento de agregar datos a un archivo cuando se ha llegado al límite de bloques duro producirá un error. También existen verificaciones similares para el número de archivos. Cuando un usuario trata de

iniciar sesión, el sistema examina el archivo de cuotas para ver si el usuario ha excedido el límite suave para el número de archivos o el número de bloques de disco. Si se ha violado uno de los límites, se muestra una advertencia y la cuenta de advertencias restantes se reduce en uno. Si la cuenta llega a cero en algún momento, el usuario ha ignorado demasiadas veces la advertencia y no se le permite iniciar sesión. Para obtener permiso de iniciar sesión otra vez tendrá que hablar con el administrador del sistema. Este método tiene la propiedad de que los usuarios pueden sobrepasar sus límites suaves durante una sesión, siempre y cuando eliminen el exceso cuando cierren su sesión. Los límites duros nunca se pueden exceder.

4.4.3 Consistencia del sistema de archivos

Otra área donde la confiabilidad es una cuestión importante es la de consistencia del sistema de archivos. Muchos sistemas de archivos leen bloques, los modifican y los escriben posteriormente. Si el sistema falla antes de escribir todos los bloques modificados, el sistema de archivos puede quedar en un estado inconsistente. Este problema es muy crítico si algunos de los bloques que no se han escrito son bloques de nodos-i, bloques de directorios o bloques que contienen la lista de bloques libres. Para lidiar con el problema de los sistemas de archivos inconsistentes, la mayoría de las computadoras tienen un programa utilitario que verifica la consistencia del sistema de archivos. Por ejemplo, UNIX tiene a fsck y Windows tiene a scandisk. Esta herramienta se puede ejecutar cada vez que se arranca el sistema, en especial después de una falla. La descripción de más adelante nos indica cómo funciona fsck. Scandisk es un poco distinto, ya que opera en un sistema de archivos diferente, pero el principio general de utilizar la redundancia inherente del sistema de archivos para repararlo todavía es válido. Todos los verificadores de sistema de archivos comprueban cada sistema de archivos (partición de disco) de manera independiente de los demás. Se pueden realizar dos tipos de verificaciones de consistencia: archivos y bloques. Para comprobar la consistencia de los bloques, el programa crea dos tablas, cada una de las cuales contiene un contador para cada bloque, que al principio se establece en 0. Los contadores en la primera tabla llevan el registro de cuántas veces está presente cada bloque en un archivo; los contadores en la segunda tabla registran con qué frecuencia está presente cada bloque en la lista de bloques libres (o en el mapa de bits de bloques libres). Después el programa lee todos los nodos-i utilizando un dispositivo puro, el cual ignora la estructura de los archivos y sólo devuelve todos los bloques de disco que empiezan en 0. Si partimos de un nodo-i, es posible construir una lista de todos los números de bloque utilizados en el archivo correspondiente. A medida que se lee cada número de bloque, se incrementa su contador en la primera tabla. Después el programa examina la lista o mapa de bits de bloques libres para encontrar todos los bloques que no estén en uso. Cada ocurrencia de un bloque en la lista de bloques libres hace que se incremente su contador en la segunda tabla.

Si el sistema de archivos es consistente, cada bloque tendrá un 1 en la primera tabla o en la segunda, como se ilustra en la figura 4-27(a). Sin embargo, como resultado de una falla, las tablas podrían verse como en la figura 4-27(b), donde el bloque 2 no ocurre en ninguna de las dos tablas. Se reportará como un bloque faltante. Aunque los bloques faltantes no hacen un daño real, desperdician espacio y por ende reducen la capacidad del disco. La

solución a los bloques faltantes es directa: el verificador del sistema de archivos sólo los agrega a la lista de bloques libres.

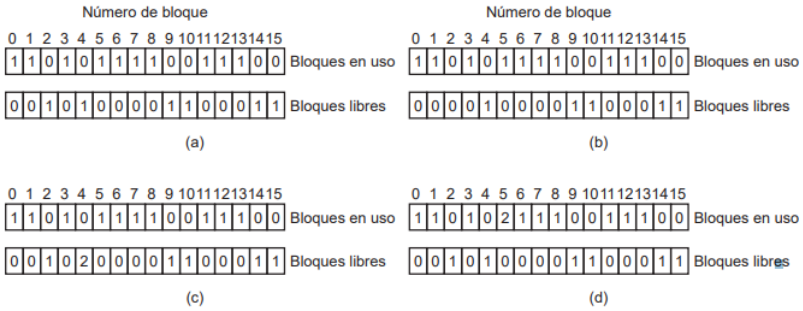

Figura 4-27. Estados del sistema de archivos. (a) Consistente. (b) Bloque faltante. (c) Bloque duplicado en la lista de bloques libres. (d) Bloque de datos duplicado.

Otra situación que podría ocurrir es la de la figura 4-27(c). Aquí podemos ver un bloque, el número 4, que ocurre dos veces en la lista de bloques libres (puede haber duplicados sólo si la lista de bloques libres es en realidad una lista; con un mapa de bits es imposible). La solución aquí también es simple: reconstruir la lista de bloques libres. Lo peor que puede ocurrir es que el mismo bloque de datos esté presente en dos o más archivos, como se muestra en la figura 4.27(d) con el bloque 5. Si se remueve alguno de esos archivos, el bloque 5 se colocará en la lista de bloques libres, lo cual producirá una situación en la que el mismo bloque, al mismo tiempo, esté en uso y sea libre. Si se remueven ambos archivos, el bloque se colocará dos veces en la lista de bloques libres. La acción apropiada que debe tomar el verificador del sistema de archivos es asignar un bloque libre, copiar el contenido del bloque 5 en él e insertar la copia en uno de los archivos. De esta forma, el contenido de información en los archivos no se modifica (aunque es muy probable que uno ellos terminen con basura), pero por lo menos la estructura del sistema de archivos se hace consistente. El error se debe reportar para permitir que el usuario inspeccione los daños. Además de verificar que cada bloque se haya contabilizado apropiadamente, el verificador del sistema de archivos también verifica el sistema de directorios. Utiliza también una tabla de contadores, pero éstos son por archivo, no por bloque. Empieza en el directorio raíz y desciende recursivamente por el árbol, inspeccionando cada directorio en el sistema de archivos. Para cada nodo-i en cada directorio, incrementa un contador para la cuenta de uso de ese archivo. Recuerde que debido a los vínculos duros, un archivo puede aparecer en dos o más directorios. Los vínculos simbólicos no cuentan y no hacen que se incremente el contador para el archivo objetivo. Cuando el verificador termina, tiene una lista indexada por número de nodo-i, que indica cuántos directorios contienen cada archivo. Después compara estos números con las cuentas de vínculos almacenadas en los mismos nodos-i. Estas cuentas empiezan en 1 cuando se crea un archivo y se incrementan cada vez que se crea un vínculo (duro) al archivo. En un sistema de archivos consistentes, ambas cuentas concordarán. Sin embargo, pueden ocurrir dos tipos de errores: que la cuenta de vínculos en el nodo-i sea demasiado alta o demasiado baja. Si la cuenta de vínculos es mayor que el número de entradas en el directorio, entonces aun si se remueven todos los archivos de los directorios, la cuenta seguirá siendo distinta de cero y el nodo-i no se removerá. Este error no es grave, pero desperdicia espacio en el disco con archivos que no están en ningún directorio. Para corregirlo, se debe establecer la cuenta de vínculos en el nodo-i al valor correcto. El otro error es potencialmente catastrófico. Si dos entradas en el directorio están vinculados a un archivo, pero el nodo-i dice que sólo hay una, cuando se elimine una de las

dos entradas del directorio, la cuenta de nodos-i será cero. Cuando una cuenta de nodos-i queda en cero, el sistema de archivos la marca como no utilizada y libera todos sus bloques. Esta acción hará que uno de los directorios apunte ahora a un nodo-i sin utilizar, cuyos bloques pueden asignarse pronto a otros archivos. De nuevo, la solución es tan sólo obligar a que la cuenta de vínculos en el nodo-i sea igual al número actual de entradas del directorio. Estas dos operaciones, verificar bloques y verificar directorios, se integran a menudo por cuestiones de eficiencia (es decir, sólo se requiere una pasada sobre los nodos-i). También son posibles otras comprobaciones. Por ejemplo, los directorios tienen un formato definido, con números de nodos-i y nombres ASCII. Si un número de nodo-i es más grande que el número de nodos-i en el disco, el directorio se ha dañado. Además, cada nodo-i tiene un modo, algunos de los cuales son legales pero extraños, como 0007, que no permite ningún tipo de acceso al propietario y su grupo, pero permite a los usuarios externos leer, escribir y ejecutar el archivo. Podría ser útil por lo menos reportar los archivos que dan a los usuarios externos más derechos que al propietario. Los directorios con más de, por decir, 1000 entradas, son también sospechosos. Los archivos localizados en directorios de usuario, pero que son propiedad del superusuario y tienen el bit SETUID activado, son problemas potenciales de seguridad debido a que dichos archivos adquieren los poderes del superusuario cuando son ejecutados por cualquier usuario. Con un poco de esfuerzo, podemos reunir una lista bastante extensa de situaciones técnicamente legales, pero aun así peculiares, que podría valer la pena reportar. En los párrafos anteriores hemos analizado el problema de proteger al usuario contra las fallas. Algunos sistemas de archivos también se preocupan por proteger al usuario contra sí mismo. Si el usuario trata de escribir

$$rm *. o$$

para quitar todos los archivos que terminen con ".o" (archivos objeto generados por el compilador), pero escribe por accidente:

$$rm *. o$$

(observe el espacio después del asterisco), rm eliminará todos los archivos en el directorio actual y después se quejará de que no puede encontrar .o. En MS-DOS y algunos otros sistemas, cuando se remueve un archivo todo lo que ocurre es que se establece un bit en el directorio o nodo-i que marca el archivo como removido. No se devuelven bloques de disco a la lista de bloques libres sino hasta que realmente se necesiten. Así, si el usuario descubre el error de inmediato, es posible ejecutar un programa de utilería especial que "des remueve" (es decir, restaura) los archivos removidos. En Windows, los archivos que se remueven se colocan en la papelera de reciclaje (un directorio especial), desde donde se pueden recuperar más tarde, si se da la necesidad. Desde luego que no se reclama el espacio de almacenamiento sino hasta que realmente se remuevan de este directorio